Ex. A, p. 1 of 6

USPTO Patent Full-Text and Image Database

Searching 1976 to present...

Results of Search in 1976 to present db for:
ACLM/"computer readable medium": 13867 patents.
Hits 1 through 50 out of 13867

Next 50 Hits

Jump To

Refine Search ACLM/"computer readable medium"

	PAT. NO.	Title
1	6,807,667	Method and system of an application program interface for abstracting network traffic control components to application programs
2	6,807,666	Methods and arrangements for providing multiple concurrent desktops and workspaces in a shared computing environment
3	6,807,664	Multiapplication concurrent-execution interpreter execution method therefor and computer-readable storage medium storing program therefor
4	6,807,662	Performance of integrated circuit components via a multiple exposure technique
5	6,807,659	Robust delay metric for RC circuits
6	6,807,657	Inter-signal proximity verification in an integrated circuit
7	6,807,656	Decoupling capacitance estimation and insertion flow for ASIC designs
8	6,807,651	Procedure for optimizing mergeability and datapath widths of data flow graphs
9	6,807,648	Variable-strength error correction in ad-hoc networks
10	6,807,646	System and method for time slicing deterministic patterns for reseeding in logic built-in self-test
11	6,807,641	Content provider system
12	6,807,634	Watermarks for customer identification
13	6,807,627	Preserving the content of a first register without affecting the content of a second register
14	6,807,623	Data processing control system, controller, data processing control method, program, and medium
15	6,807,618	Address translation
16	6,807,616	Memory address checking in a proccesor that support both a segmented and a

16. The method of claim 1 further comprising the steps of:

receiving a query traffic control message having the interface identification from the application program as a request to obtain a traffic control parameter for the opened interface;

querying the traffic control component to obtain the traffic control parameter of the opened interface identified by the interface identification responsive to the query traffic control message;

deciphering response data including the traffic control parameter from the traffic control component; and

providing the deciphered response data with the traffic control parameter to the application program.

17. The method of claim 16 wherein the query traffic control message includes a notify change boolean as a request to notify the application program whenever the traffic control parameter is changed in the network.

18. The method of claim 1 further comprising the steps of:

receiving a set traffic control message having the interface identification from the application program as a request to set a traffic control parameter for the opened interface; and

communicating a control instruction to the traffic control component to set the traffic control parameter for the opened interface identified by the interface identification.

19. The method of claim 1 further comprising the steps of:

receiving a deregistration message having the client identification from the application program as a request to deregister from the network; and

deleting the saved client identification responsive to the deregistration message.

20. The method of claim 1 wherein the network traffic component includes:

a packet scheduler for scheduling data transmissions through the network connection, wherein the packet scheduler communicates using windows management instrumentation messages; and

a generic packet classifier for classifying data transmissions through the network connection, wherein the generic packet classifier communicates using input/output control messages.

21. A computer-readable medium having computer-executable instructions for performing the steps of:

receiving a registration message from an application program as a request to establish a connection with a network;

assigning a client identification to the application program responsive to the registration message;

saving the client identification to identify the application program for subsequent calls;

receiving an open interface message from the application program as a request to open an interface that controls network traffic;

assigning an interface identification to the interface opened in response to the open interface message;

saving the interface identification to identify the interface for subsequent calls;

communicating a control instruction to a traffic control component to open the interface; and

returning the interface identification to the application program to indicate that the interface has been opened.

22. The computer-readable medium of claim 21 having further computer-executable instructions for performing the steps of:

receiving a response data message from the traffic control component;

deciphering the response data message in a format that the application program can interpret; and

providing the deciphered response data to the application program.

23. The computer-readable medium of claim 21 having further computer-executable instructions for performing the steps of:

receiving a flow install message having the interface identification from the application program as a request to install a flow to the opened interface;

assigning a flow identification to the flow responsive to the flow install message;

saving the flow identification to identify the flow for subsequent calls; and

communicating a control instruction to the traffic control component to install the flow to the opened interface; and

providing the flow identification to the application program to indicate that the flow has been installed to the open interface.

24. The computer-readable medium of claim 23 having further computer-executable instructions for performing the steps of:

receiving a modify flow message having the flow identification from the application program as a request to modify the flow of the opened interface; and

communicating a control instruction to the traffic control component to delete the flow identified by the flow identification.

25. The computer-readable medium of claim 23 having further computer-executable instructions for performing the steps of:

receiving a delete flow message having the flow identification from the application program as a request to delete the flow of the opened interface;

communicating a control instruction to the traffic control component to delete the flow identified by the flow identification.

26. The computer-readable medium of claim 23 having further computer-executable instructions for performing the steps of:

receiving a registration message from a second application program as a request to register with the network;

assigning a second client identification to the second application program;

receiving an open interface message from the second application program as a request to open the interface assigned with the interface identification;

providing the interface identification to the second application program to indicate that the interface has been opened;

receiving a flow install message having the interface identification from the second application program as a request to install a second flow to the opened interface;

assigning a second flow identification to the second flow responsive to the flow install message; and

communicating a control instruction to the traffic control component to aggregate the second flow with the first flow from the first application program, on the interface assigned with the interface identification.

27. The computer-readable medium of claim 23 having further computer-executable instructions for performing the steps of:

Returning to decision step **206**, if a desktop is already open for the user as demonstrated by an associated entry in ODL **208**, then step **206** (YES) advances process **200** to step **224**. In step **224**, a switch is made to the existing desktop thread, followed by a switch to the applicable desktop **222**, in step **220**. In FIG. **3** there are N>1 desktops as represented by desktops **222***a–c*.

Reference is now made to FIG. **4**, wherein a flowchart depicts a process **240** for logging off of a desktop **222**. In this example, there are three ways for the user to become "logged off". The first takes place in step **242**, wherein the user intentionally and fully logs off. When such a logoff event is generated in step **242**, then in step **244** the desktop is cleaned up, ODL **208** is updated and the thread is terminated. After step **244**, process **240** returns to logon screen **100**.

The second way for a user to become "logged off" is provided in step **246**, wherein the user switches desktops manually. At any point prior to logging off, the user can initiate an action (e.g., by user input), which will return them to logon screen **100** without signaling the user's thread that a logoff has occurred. When this happens, the users desktop remains running and all applications that have been started on that desktop continue to run. However, the user is unable to see any of these applications because their desktop is essentially hidden. At this point, the desktop is said to be "switched out", but it is still active. The user will need to log on again in accord with process **200** to have the active desktop "switched in".

The third way that a user can become "logged off" is shown in step **248**, wherein after a user configurable period of inactivity (e.g., no mouse activity or keyboard input) the operating system will assume that the user has stopped working on the computer for the time being and it will automatically switch back to logon screen **100**. This helps to improve security by securing the user's desktop when they no longer appear to be using the computer. It also makes it easier for other users to utilize the computer and not worry about ending another user's session.

It is also possible to provide a plurality of separate and concurrent environments within the shared computing environment by switching remote processes, such as, for example, WinStations, instead of desktops. Logically, WinStations contain multiple desktops. Creating a WinStation for each user and switching the active WinStation when a user logs on or switches back to logon screen **100** could implement the same mechanism as described above.

Although some preferred embodiments of the various methods and arrangements of the present invention have been illustrated in the accompanying Drawings and described in the foregoing Detailed Description, it will be understood that the invention is not limited to the exemplary embodiments disclosed, but is capable of numerous rearrangements, modifications and substitutions without departing from the spirit of the invention as set forth and defined by the following claims.

What is claimed is:

1. A method comprising:

configuring a single computer to be concurrently and physically shared by multiple users by executing a plurality of concurrent switchable desktop environments within the single computer by:

presenting a logon user interface to each said user physically seeking to use said computer; and

within the computer:

creating a separate desktop thread for each said user that is authenticated by the logon user interface;

creating a separate desktop associated with each said desktop thread for said concurrent switchable desktop environments;

displaying only one said desktop as active at a time; and

maintaining a list of said desktop threads to support switching from a first said desktop to a second said desktop.

2. The method as recited in claim **1**, further comprising:

establishing a separate user environment associated with each said desktop.

3. The method as recited in claim **1**, further comprising:

launching a separate user shell associated with each said desktop.

4. The method as recited in claim **1**, further comprising:

selectively switching from the first said desktop to the second said desktop without terminating the desktop thread associated with the first said desktop.

5. The method as recited in claim **1**, further comprising:

automatically switching from a first said desktop to a second said desktop without terminating the desktop thread associated with the first said desktop; and

launching a separate user shell associated with each said desktop.

6. The method as recited in claim **5**, wherein the automatically switching from the first said desktop to the second said desktop occurs following a defined period of said user inactivity.

7. The method as recited in claim **1**, further comprising:

selectively removing at least one said desktop thread from the list of said desktop threads when said user logs off.

8. A computer-readable medium having computer executable instructions for causing at: least one processor to perform steps comprising:

configuring a single computer to be concurrently and physically shared by multiple users by executing a plurality of concurrent switchable desktop environments within the single computer by:

presenting a logon user interface to each said user physically seeking to use said computer; and

within the computer:

creating a separate desktop thread for each said user that is authenticated by the logon user interface;

creating a separate desktop associated with each said desktop thread for said concurrent switchable desktop environments;

displaying only one said desktop as active at a time; and

maintaining a list of said desktop threads to support switching from a first said desktop to a second said desktop.

9. The computer-readable medium as recited in claim **8**, having further computer-executable instructions for performing the step of:

establishing a separate user environment associated with each said desktop.

10. The computer-readable medium as recited in claim **8**, having further computer-executable instructions for performing the step of:

launching a separate user shell associated with each said desktop.

11. The computer-readable medium as recited in claim **8**, having further computer-executable instructions for performing the step of:

selectively switching from the first said desktop to the second said desktop without terminating the desktop thread associated with the first said desktop.

31. A method as claimed in claim 30, wherein the class-retrieval path is stored in the shared area.

32. A method as claimed in claim 29, wherein the specifying information also specifies class-retrieval paths together with the application, and said allocating step stores the specified class-retrieval paths in an individual application area corresponding to the application.

33. A method as claimed in claim 29, wherein said loading step judges if a new class for which no bytecodes have been loaded is required for execution of individual commands of the applications, and if the new class is judged to be required, said loading step loads the bytecodes for the new class in the shared area.

34. A method as claimed in claim 29, further comprising:

a counting step of, for individual classes, indicating the number of applications using the classes;

a decrement step of decreasing a value of the counter for the class used by the application when an application is terminated; and

an unloading step of unloading bytecodes for a class for which the value of the counter is zero.

35. A method as claimed in claim 28, wherein said allocating step judges whether or not the new class has a class variable by analyzing the bytecodes of the class, which are stored in the shared area.

36. A method as claimed in claim 28, further comprising an initializing step of analyzing the bytecodes to judge if the class variable of the new class must be initialized, and for initializing the class variable of the new class if the class variable must be initialized.

37. A method as claimed in claim 36, wherein said initializing step judges the class variable to be initialized if the bytecodes include initial data for the class variable, and uses the initial data to initialize the class variable.

38. A method as claimed in claim 36, wherein said initializing step judges the class variable to be initialized if the bytecodes include a class-variable initializing method, and executes the class-variable initializing method to initialize the class variable.

39. A computer-readable program stored on a computer readable medium for controlling a computer to perform functions of an interpreter for a programming language supporting class variables, the program comprising codes for instructing the computer to perform a method comprising:

a receiving step of receiving specifying information for specifying an application to be started;

a generating step of generating an individual application area corresponding to the application specified by the specifying information, wherein the individual application area does not store the data to be shared by a plurality of applications;

a loading step of loading in a shared area, data for a class required to execute the specified application and to be shared by a plurality of applications;

an allocating step of allocating class variable areas for the class variables in the individual application area corresponding to the specified application if a new class having class variables is detected from classes required to execute the specified application;

an executing step of executing the specified application by accessing the class variables in the individual application area corresponding to the specified application when the class variables are to be accessed;

a deleting step of deleting the individual application area corresponding to the specified application when the specified application is terminated;

a determination step of determining when the specified application is terminated, whether or not the class required to execute the specified application is used by any other applications; and

an unloading step of unloading from the shared area, data for the class required to execute the specified application if the class is determined not to be used by any other applications.

* * * * *

Ex. A, p. 5 of 6

52. The set of masks of claim 51 wherein the definitions are created further by adding a clear area corresponding to any of the one or more assist features within the dark feature of the second revised layout definition when the width of the any of the one or more assist features exceeds a predefined threshold.

53. An integrated device layer, created by exposing a photoresist applied on a wafer to a first mask created by a first layout definition and thereafter to a second mask created by a second layout definition, and processing the photoresist, wherein:

the first layout definition comprises two or more target features from an initial layout of the integrated device layer that are separated by a distance that is less than a limit associated with isolated patterns and are surrounded by clear areas inside dark-field content; and

the second layout definition comprises one or more dark features inside bright-field content, the one or more dark features overlapping the two or more target features and at least a portion of the corresponding clear areas when being used in a multiple exposure fabrication process.

54. The integrated device layer of claim 53 wherein each of the first mask and the second mask is any one of a binary mask and an attenuated phase-shifting mask.

55. The integrated device layer of claim 54 wherein the dark feature of the second layout definition covers an area beginning with a leftmost clear area adjacent to a first target feature of the two target features and ending with the rightmost clear area adjacent to a second target feature of the two target features.

56. The integrated device layer of claim 54 wherein the distance is between the first limit associated with the isolated pattern and a second limit associated with a dense pattern.

57. The integrated device layer of claim 54 wherein the first layout definition further includes one or more assist features between the two target features.

58. The integrated device layer of claim 54 the second layout definition further includes a clear area corresponding to any of the one or more assist features within the dark feature of the second layout definition if the width of the any of the one or more assist features exceeds a predefined threshold.

59. The integrated device layer of claim 53 wherein each of the one or more target features corresponds to a gate of a transistor.

60. A computer-readable medium comprising executable instructions which when executed on a processing system cause said processing system to perform a method comprising:

reading at least a portion of an initial layout of an integrated device layer; identifying two or more target features within the initial layout separated by a distance that is less than a limit associated with isolated patterns;

creating a first revised layout definition associated with a first mask, the first revised layout definition including the one or more target features surrounded by clear areas inside dark-field content; and

creating a second revised layout definition associated with a second mask, the second revised layout definition including one or more dark features inside bright-field content, wherein the one or more dark features, when used in the multiple exposure fabrication process, will overlap the two or more target features and at least a portion of the corresponding clear areas.

61. A computer-readable medium on which data is stored for defining a first and second mask for creating at least one target feature defined in an initial layout of an integrated circuit device with a multiple exposure fabrication, the medium including:

data representing a first mask including a dark feature and clear areas adjoining the dark feature within a dark-field that correspond to the target feature in the initial layout; and

data representing a second mask including a feature within a bright-field, wherein the feature of the second mask covers the dark feature and at least a portion of the clear areas adjoining the dark feature when used in a multiple exposure process.

62. The computer-readable medium of claim 61, wherein the data representing the first mask further includes data representing a second dark feature and clear areas adjoining the second dark feature that corresponds to a second target feature that is within a predetermined distance of the first target feature, wherein the feature in the second mask covers both dark features and at least a portion of the adjoining clear areas in the first mask when used in a multiple exposure process.

63. The computer-readable medium of claim 62, wherein the data for the first mask further includes data representing one or more assist features that are between the first and second dark features that correspond to the first and second target features.

64. The computer-readable medium of claim 63, wherein the feature in the second mask includes a bright-field opening aligned with the one or more assist features on the first mask.

65. A method for separating an initial layout of an integrated circuit device into a set of layout definitions for use in a multiple exposure fabrication process, comprising:

reading at least a portion of the initial layout;

identifying one or more target features within the initial layout that require fabrication with a multiple exposure fabrication process;

creating a first revised layout definition for a first mask; the first revised layout definition including dark features with adjacent clear areas inside a dark-field that correspond to each of the target features requiring fabrication with a multiple exposure process; and

creating a second revised layout definition for a second mask, the second revised layout definition including dark features in a bright-field that overlap the dark features and at least a portion of the adjacent clear areas that correspond to the target features in the first mask when used in a multiple exposure fabrication process and dark features in a bright-field that correspond to features from the initial layout that do not require a multiple exposure fabrication process.

66. A computer-readable medium of which data is stored for creating a first and second mask to create a layout of an integrated circuit device with a multiple exposure process, the medium containing:

data representing dark features and adjacent clear areas within a dark-field that correspond to target features within the layout requiring fabrication with a multiple exposure process; and

data representing dark features within a clear field that overlap the dark features and at least a portion of the adjacent clear areas corresponding to the target features and data representing dark features within the clear field that correspond to features in the layout of the integrated circuit that do not require fabrication with a multiple exposure process.

* * * * *

$1-e^{-(t/\beta)\Gamma(1,\theta)}=\phi$

which yields the solution

$t_\phi=\beta\cdot\{1n[1/(1-\phi)]\}^\theta$.

In particular, the 50% delay point can be calculated as

$t_{0.5}=\beta\cdot[1n(2)]^\theta\approx 0.693^\theta$.

The foregoing implementation of the present invention maybe further understood with reference to the flow chart of FIG. 4. The first step is to calculate the two circuit impulse response moments m_1 and m_2 (60). The ratio r is then calculated from these moments, as $r=m_2/m_1^2$ (62). A table look-up (Table 1) is used to find θ from r (64). Another table look-up (Table 2) is used to find $\Gamma(1+\theta)$ (66). The distribution parameter is then set to $\beta=-m_1/\Gamma(1+\theta)$ (68). The final step returns the appropriate delay value, e.g., a 50% delay value of $\beta\cdot 0.693^\theta$(70).

Empirical observations support the conclusion that θ can be used as a resistive shielding factor, to provide a quantitative description of the degree of resistive shielding. A resistive shielding factor of larger than 1 indicates that the node is strongly shielded, while a shielding factor of less than 1 indicates that the node is under shielded.

Some delay metrics are not guaranteed to return non-negative real values. A Weibull distribution is stable when both α and β are positive. For an RC circuit the second impulse response moment is always positive, so θ is guaranteed to be positive for any node in the circuit. Also, for an RC circuit, the first impulse response moment is always negative, so the gamma function is always positive, as long as θ is positive. Accordingly, the Weibull-based delay metric will always return a nonnegative value for delay for any RC circuit.

This technique is quite robust and has satisfactory accuracies at both near- and far-end nodes. The invention is also straightforward to implement, and very efficient, using only two small lookup tables, one having 15 entries, and the other having 11 entries. Experiments have shown that the present invention provides satisfactory results when prior art approaches, such as PRIMO, might yield unrealistic results.

Although the invention has been described with reference to specific embodiments, this description is not meant to be construed in a limiting sense. Various modifications of the disclosed embodiments, as well as alternative embodiments of the invention, will become apparent to persons skilled in the art upon reference to the description of the invention. It is therefore contemplated that such modifications can be made without departing from the spirit or scope of the present invention as defined in the appended claims.

What is claimed is:

1. A method of estimating delays at nodes in an RC circuit, comprising the steps of:

calculating a first impulse response moment and a second impulse response moment of the RC circuit; and

matching the first and second impulse response moments to a Weibull distribution, wherein said matching step includes the step of computing a signal delay value at a delay point by finding a percentile of the Weibull distribution corresponding to the delay point.

2. The method of claim 1, further comprising the step of determining whether the RC circuit meets a desired optimization condition based on the signal delay value.

3. The method of claim 1 wherein said computing step uses no more than two table look-ups.

4. The method of claim 3 wherein said computing step further includes the step of finding a resistive shielding factor from a first table.

5. The method of claim 4 wherein said computing step further includes the step of evaluating a gamma function using a second table.

6. A data processing system for estimating delays at nodes in an RC circuit, comprising:

means for processing program instructions;

a memory device connected to said processing means; and

program instructions residing in said memory device for calculating a first impulse response moment and a second impulse response moment of the RC circuit, and matching the first and second impulse response moments to a Weibull distribution, wherein the matching of the first and second impulse response moments includes computing a signal delay value at a delay point by finding a percentile of the Weibull distribution corresponding to the delay point.

7. The data processing system of claim 6, wherein said program instructions further determine whether the RC circuit meets a desired optimization condition based on the signal delay value.

8. The data processing system of claim 6, wherein said program instructions use no more than two table look-ups for the computing of the signal delay.

9. The data processing system of claim 8, wherein said program instructions further find a resistive shielding factor from a first table.

10. The data processing system of claim 9, wherein said program instructions further evaluate a gamma function using a second table.

11. A computer program product for estimating delays at nodes in an RC circuit, comprising:

a computer-readable medium; and

program instructions residing in said medium for calculating a first impulse response moment and a second impulse response moment of the RC circuit, and matching the first and second impulse response moments to a Weibull distribution, wherein the matching of the first and second impulse response moments includes computing a signal delay value at a delay point by finding a percentile of the Weibull distribution corresponding to the delay point.

12. The computer program product of claim 11, wherein said program instructions further determine whether the RC circuit meets a desired optimization condition based on the signal delay value.

13. The computer program product of claim 11, wherein said program instructions use no more than two table look-ups for the computing of the signal delay.

14. The computer program product of claim 13 wherein said program instructions further find a resistive shielding factor from a first table.

15. The computer program product of claim 14 wherein said program instructions further evaluate a gamma function using a second table.

* * * * *